**Create authentication service that returns JWT**   
  
  
  
**Request**

curl -s -u user:pwd http://localhost:8090/authenticate

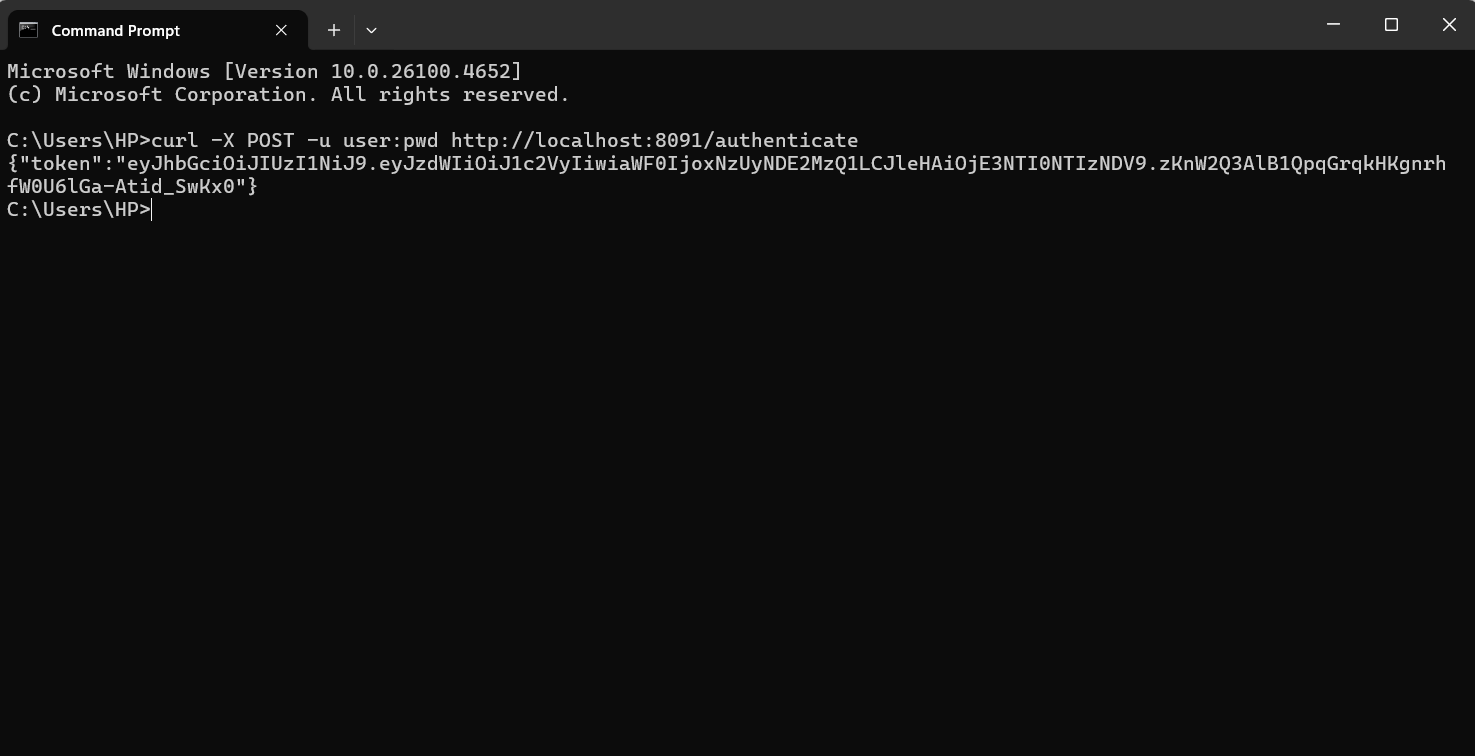
**Response**

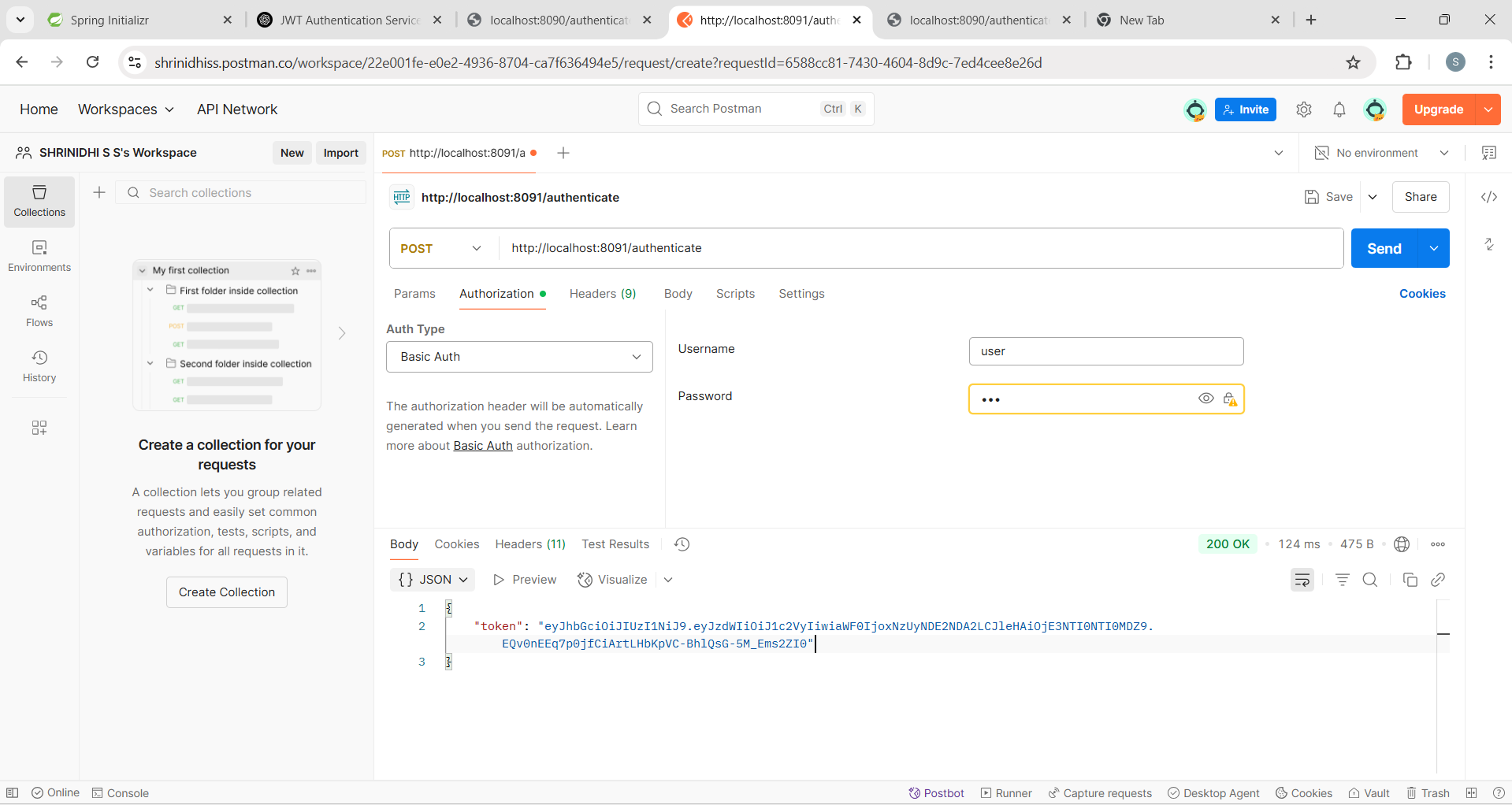
{"token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyIiwiaWF0IjoxNTcwMzc5NDc0LCJleHAiOjE1NzAzODA2NzR9.t3LRvlCV-hwKfoqZYlaVQqEUiBloWcWn0ft3tgv0dL0"}

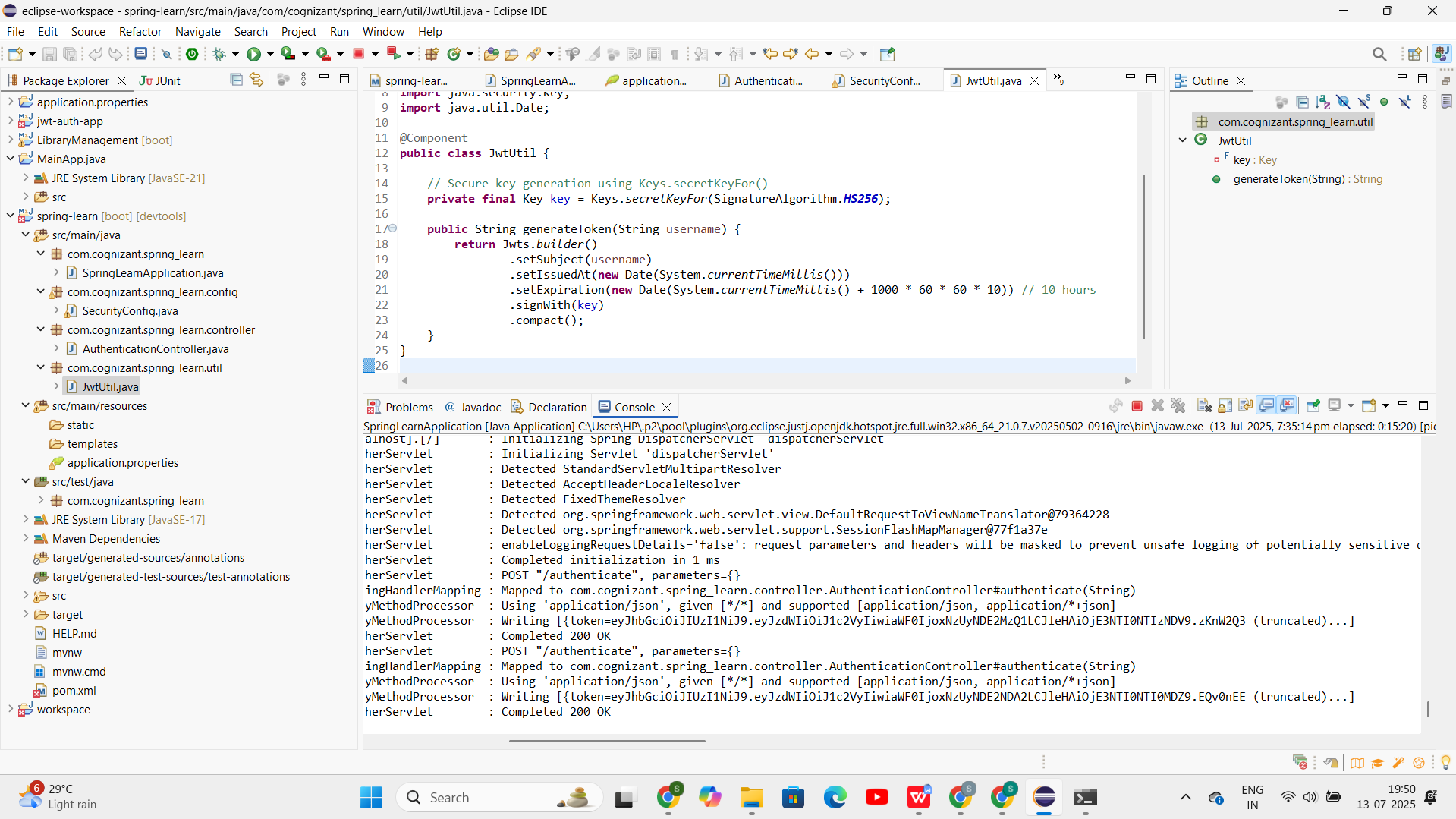
This can be incorporated as three major steps:

* Create authentication controller and configure it in SecurityConfig
* Read Authorization header and decode the username and password
* Generate token based on the user retrieved in the previous step

Let incorporate the above as separate hands on exercises.







**SECURITYCONFIG.JAVA**

**package** com.cognizant.spring\_learn.config;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.Configuration;

**import** org.springframework.security.authentication.AuthenticationManager;

**import** org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfiguration;

**import** org.springframework.security.config.annotation.web.builders.HttpSecurity;

**import** org.springframework.security.config.annotation.web.configurers.AbstractHttpConfigurer;

**import** org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

**import** org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

**import** org.springframework.security.crypto.password.PasswordEncoder;

**import** org.springframework.security.provisioning.InMemoryUserDetailsManager;

**import** org.springframework.security.core.userdetails.User;

**import** org.springframework.security.core.userdetails.UserDetailsService;

**import** org.springframework.security.web.SecurityFilterChain;

@Configuration

@EnableWebSecurity

**public** **class** SecurityConfig {

@Bean

**public** SecurityFilterChain securityFilterChain(HttpSecurity http) **throws** Exception {

http

.csrf(AbstractHttpConfigurer::disable)

.authorizeHttpRequests(auth -> auth

.requestMatchers("/authenticate").permitAll()

.anyRequest().authenticated()

)

.~~httpBasic~~();

**return** http.build();

}

@Bean

**public** UserDetailsService userDetailsService(PasswordEncoder passwordEncoder) {

**return** **new** InMemoryUserDetailsManager(

User.*builder*()

.username("user")

.password(passwordEncoder.encode("pwd"))

.roles("USER")

.build()

);

}

@Bean

**public** PasswordEncoder passwordEncoder() {

**return** **new** BCryptPasswordEncoder();

}

@Bean

**public** AuthenticationManager authenticationManager(AuthenticationConfiguration config) **throws** Exception {

**return** config.getAuthenticationManager();

}

}

**AuthenticationController.java**

**package** com.cognizant.spring\_learn.controller;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.security.authentication.AuthenticationManager;

**import** org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

**import** org.springframework.web.bind.annotation.PostMapping;

**import** org.springframework.web.bind.annotation.RequestHeader;

**import** org.springframework.web.bind.annotation.RestController;

**import** java.nio.charset.StandardCharsets;

**import** java.util.Base64;

**import** java.util.Collections;

**import** com.cognizant.spring\_learn.util.JwtUtil;

@RestController

**public** **class** AuthenticationController {

@Autowired

**private** AuthenticationManager authenticationManager;

@Autowired

**private** JwtUtil jwtUtil;

@PostMapping("/authenticate")

**public** ResponseEntity<?> authenticate(@RequestHeader("Authorization") String authHeader) {

String base64Credentials = authHeader.substring("Basic ".length());

**byte**[] credDecoded = Base64.*getDecoder*().decode(base64Credentials);

String credentials = **new** String(credDecoded, StandardCharsets.***UTF\_8***);

String[] values = credentials.split(":", 2);

String username = values[0];

String password = values[1];

**try** {

authenticationManager.authenticate(**new** UsernamePasswordAuthenticationToken(username, password));

} **catch** (Exception e) {

**return** ResponseEntity.*status*(HttpStatus.***UNAUTHORIZED***).body("Invalid credentials");

}

String token = jwtUtil.generateToken(username);

**return** ResponseEntity.*ok*(Collections.*singletonMap*("token", token));

}

}